# OPTIMIZAREA INTEROGĂRILOR DISTRIBUITE

**Nicoleta IACOB,** *Student Doctorand,*
*Universitatea din Piteşti*

***REZUMAT:*** *Necesitatea sistemelor distribuite a fost determinată de tipul de afaceri dezvoltate de către companii cu sedii distribuite geografic unde structura organizaţională specifică promovează un model de afaceri descentralizat. Această lucrare descrie tehnicile şi conceptele de arhitectură ale unui sistem distribuit de gestionare a bazelor de date, urmată de prezentarea etapelor de implementare implicate atunci când se ocupă cu interogări distribuite în sisteme distribuite. Scopul optimizării interogărilor este de a determina cel mai eficient mod de a executa o interogare într-un mediu distribuit, atât prin obţinerea unui timp de răspuns al sistemului cât mai mic cât şi prin minimizarea timpului de execuţie a interogărilor. Pentru aceasta, vom studia factorii care influenţează modalităţile de execuţie a interogărilor şi vom analiza, de asemenea strategiile disponibile de optimizare a execuţiei interogărilor.*

***CUVINTE CHEIE****: arhitectura, interogări distribuite, optimizare, strategii.*

## 1. Introducere

O **bază de date distribuită (BDD)** este o colecţie de date interconectate logic, care sunt fizic distribuite pe staţii (site-uri) într-o reţea. Fiecare staţie din reţea are autonomie de procesare a aplicaţiilor locale, aplicaţii care sunt executate în întregime pe acea staţie. De asemenea, fiecare staţie participă la faza de execuţie a cel puţin unei aplicaţii globale, care necesită accesarea datelor de la mai multe staţii.

Un **sistem de gestiune a bazelor de date distribuite (SGBDD)** este un sistem de programe care permite gestionarea bazei de date distribuite, permiţând utilizatorilor accesul transparent la informaţii dispersate în întreaga reţea.

## 2. Arhitectura unui SGBDD

# DISTRIBUTED QUERY OPTIMIZATION

**Nicoleta IACOB**, *PhD Student,*
*University of Piteşti*

***ABSTRACT:*** *The need for the distributed systems has been determined by the type of business developed by companies with offices geographically distributed where the specific organizational structure promotes a decentralized business model. This paper describes the techniques and concepts of system architecture for distributed database management systems, followed by the presentation of implementation phases involved when dealing with the distributed queries across distributed systems. The goal of query optimization is to determine the most efficient way to execute a query in a distributed environment, by obtaining a lower system response time and also by minimizing the query execution time. For this, we will analyze the factors that influence the ways to execute a query and we will also review the available strategies to optimize the distributed query execution.*

***KEY-WORDS****: architecture, distributed queries, optimization, strategies.*

## 1. Introduction

A **distributed database (DDB)** is a collection of logically interrelated data which are physically distributed on computers (nodes) over a network. Every computer in the network has the autonomy to process local applications. Also, each computer participates in the execution stage of at least one global application that requires accessing data from multiple computers.

A **distributed database management system (DDBMS)** is a layer of software, implemented on top of existing database management systems, allowing users transparent access to information dispersed across the network.

Spre deosebire de SGBD centralizate unde există o arhitectură general acceptată, în cazul SGBDD din cauza diversităţii este dificil de prezentat o astfel de arhitectură. Una dintre cele mai cunoscute arhitecturi pentru SGBDD este arhitectura ANSI-SPARC (American National Standards Institute - Standards Planning And Requirements Committee) - fiind un standard de proiectare abstract pentru un SGBDD. Cele mai multe SGBD-uri comerciale moderne sunt bazate pe acest sistem. Această arhitectură multinivel, reprezentată în figura 1, realizează obiectivul transparenţei distribuţiei.

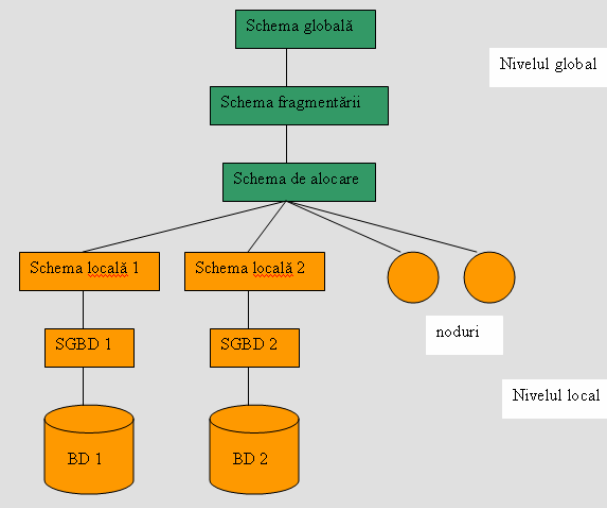


Figura 1. Arhitectura de referinţă a unui SGBDD

*Nivelul global* permite integrarea bazelor de date locale, într-o bază de date globală prin utilizarea schemei globale, schemei de fragmentare şi schemei de alocare.

*Schema globală* defineşte toate informaţiile conţinute într-o bază de date distribuită în reţea şi este descrisă de un set de relaţii globale. Fiecare relaţie globală poate fi împărţită în mai multe părţi disjuncte numite *fragmente.*

*Schema de fragmentare* descrie legăturile dintre relaţia globală şi fragmentele sale. Aceasta este de tipul unu-la-mai-mulţi şi are forma unei ierarhii. Fragmentele sunt

## 2. DDBMS architecture

Unlike centralized DBMS where there is a general accepted architecture, because of the diversity of DDBMS such an architecture is difficult to present. One of the well known architectures for DDBMS is the ANSI-SPARC Architecture (American National Standards Institute, Standards Planning And Requirements Committee) - an abstract design standard for a DDBMS. Most modern commercial DBMS are based on this system. This multilevel architecture which satisfies the distribution transparence objective is represented in the figure 1 below:

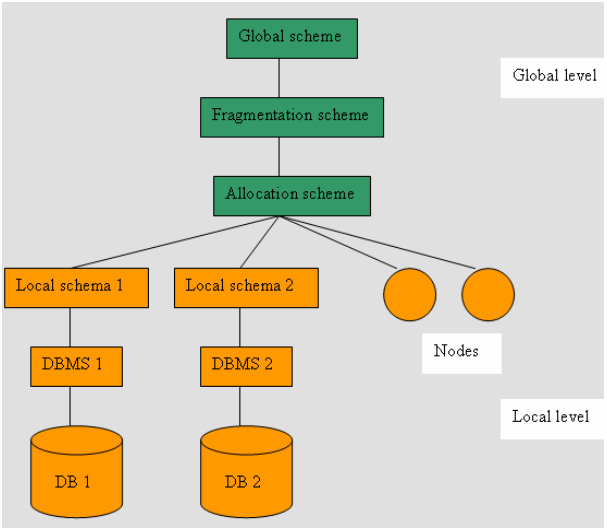


Figure 1. DDBMS reference architecture

*The global level* permits the integration of local databases in a global database by using the global scheme, the fragmentation scheme and the allocation scheme.

The *global scheme* defines all the information contained in a network distributed database and is described by a set of global relations. Every global relation can be split in many disjunctive parts called *fragments.*

The *fragmentation scheme* describes the connections between the global relation and its fragments. This is a one-to-many type

porţiuni logice ale relaţiilor globale, care pot fi alocate fizic pe unul sau mai multe noduri ale reţelei.

*Schema de alocare* descrie modul de distribuire a segmentelor pe nodurile din reţea. Fiecare segment va avea o alocare fizică pe unul sau mai multe noduri. Schema de alocare introduce o redundanţă minimă şi controlată, astfel încât un anumit segment poate fi amplasat pe mai multe noduri din reţea.

*Nivelul local* tratează toate bazele de date locale ca o bază de date centralizată. La acest nivel, *schema locală*, care depinde de tipul SGBD-ului local, realizează corespondenţa între relaţiile globale fizice de pe acel nod şi obiectele manipulate de SGBD-ul local.

### 3. Interogări distribuite

O bază de date relaţională constă din mai multe părţi, dar ea are două componente majore: *motorul de stocare* şi *procesorul de interogare*. Motorul de stocare a datelor scrie şi citeşte date de pe disc. El gestionează înregistrările, controlul concurenţei şi menţine fişierele de tip log. Procesorul de interogări acceptă sintaxa SQL, selectează un plan pentru executarea sintaxei şi execută apoi planul ales. Utilizatorul sau programul interacţionează cu procesorul de interogare, iar procesorul de interogare la rândul lui interacţionează cu motorul de stocare.

În sistemele distribuite, obiectivul execuţiei cererilor este acela de a transforma o cerere globală adresată bazei de date, văzută ca un întreg de către utilizatorii ei, în comenzi de manipulare a datelor de nivel scăzut, adresate bazelor de date locale, aflate la noduri, cu ajutorul unei strategii de execuţie eficientă. Execuţia unei cereri distribuite este compusă din mai multe **etape de execuţie**, care sunt menţionate în figura 2 de mai jos:

of scheme and has the form of a hierarchy. The fragments are logical parts of the global relations that can be physically allocated on one or more nodes of the network.

The *allocation scheme* describes the distribution mode of the segments on the nodes of the network. Every segment will have a physical allocation on one or more nodes. The allocation scheme introduces a minimum controlled redundancy, so that a certain segment can be located on multiple nodes of the network.

*The local level* treats every local database like a centralized database. At this level, the *local scheme*, which depends on local DBMS, makes the correspondence between the physical global relations on that node and the objects manipulated by the local DBMS.

### 3. Distributed queries

A relational database consists of many parts, but it has two major components: the storage engine and the query processor. The storage engine writes data to and reads data from the disk. It manages records, controls concurrency, and maintains log files. The query processor accepts SQL syntax, selects a plan for executing the syntax, and then executes the chosen plan. The user or program interacts with the query processor, and the query processor in turn interacts with the storage engine.

In distributed systems, the objective of query execution is to transform a global request in low level data manipulation commands addressed to local databases which are hosted on the nodes of the network using an efficient execution strategy. The execution of a distributed request is composed of many execution steps, which are mentioned in figure 2 below:

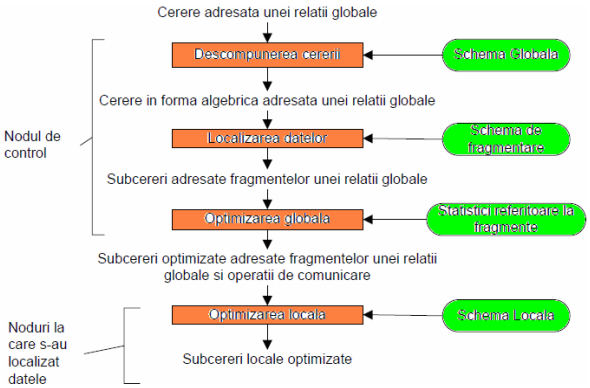


Figura 2. Schema generică de execuţie a unei cereri distribuite

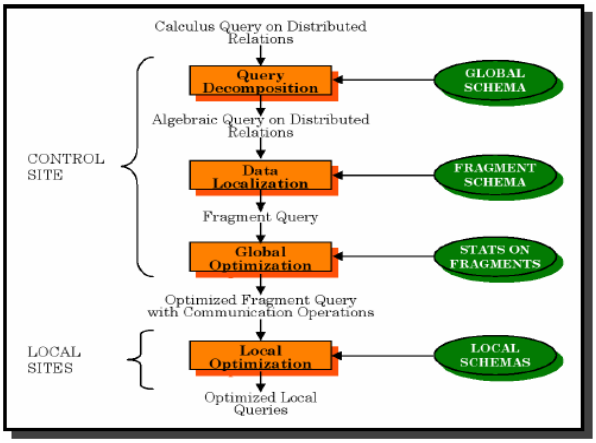


Figure 2. The general scheme for the execution of a distributed query

- Execuţia unei interogări începe cu cererea globală exprimată folosind operatorii de calcul relaţional. Această cerere se adresează unor relaţii globale, distribuirea datelelor fiind invizibilă la acest nivel. Cererea este descompusă în subcereri exprimate cu ajutorul operatorilor algebrei relaţionale. Subcererile se transformă din subcereri globale în cereri adresate fragmentelor relaţiilor globale, folosind informaţii despre schema de fragmentare.
- Optimizarea globală încearcă să găsească cea mai bună ordine a operaţiilor în cadrul subcererilor adresate fragmentelor, incluzând şi operaţiile de comunicaţie care minimizează funcţia cost.
- Optimizarea locală a fiecărei subcereri la un anumit nod se revizuieşte folosind informaţiile din schema locală.

Aceste patru etape realizează descompunerea cererii globale într-o secvenţă de operaţii locale optimizate, fiecare dintre acestea acţionând asupra unei baze de date locale. Primele trei sunt efectuate la un nod de control, care deţine informaţii despre schema globală, ultima etapă fiind efectuată la nodurile unde sunt situate datele necesare procesării cererii.

**4. Tehnici de reprezentare a interogărilor**

- The execution of a request begins with the global request represented using the relational computations. This request is addressed to some global relations, data distribution being invisible at this level. This request is decomposed in subrequests expressed with relational algebra operators. These subrequests change from global subrequests in requests addressed to the global relations fragments, using information about the fragmentation scheme.
- The global optimization tries to find the best order of operations within subrequests addressed to the fragments, including the communication operations that minimize cost function.
- Local optimization of every subrequest at a certain node is reviewed using information from local scheme.

These four phases realize decomposition of the global request into a sequence of local optimized operations, each of them acting on a local database. The first three are performed at a control node, which holds information about the global scheme, the last phase being performed at the nodes where the data needed to process the request are located.

**4. Representation techniques of**

O cerere globală poate fi exprimată folosind limbaje diferite bazate pe algebra relațională sau pe calculul relațional.

**a). Reprezentarea arborescent** este una dintre tehnicile cele mai utilizate pentru a reprezenta o expresie relațională algebrică. Arborele cererii este o reprezentare grafică a operațiilor şi operanzilor care apar într-o expresie algebrică. Relațiile (operanzii) sunt reprezentate în nodurile frunză (fiecare nod este o operație cu unul sau mai mulți operanzi) iar operațiile în nodurile interne. În execuția interogării, un nod intern poate fi executat atunci când operanzii săi sunt disponibili. După execuție, nodul se înlocuieşte cu rezultatul operației. Nodul rădăcină este ultimul executat şi înlocuit cu rezultatul final.

**b). Graful cererii** va înlocui arborele cererii dacă este folosit calculul relațional. Constantele şi atributele relației sunt reprezentate în nodurile grafului. Este permis să se repete atributele care apar în mai mult de o relație. Condițiile de joncțiune sunt reprezentate pe rândurile care leagă nodurile atribut, iar condițiile de selecție sunt reprezentate pe rândurile care leagă o constantă de un atribut.

## 5. Optimizarea interogărilor

Conceptual, optimizarea interogărilor constă din alegerea celei mai bune strategii (ordine a operațiilor) din toate strategiile posibile.

În strânsă legătură cu optimizarea interogărilor distribuite sunt conceptele de *fragmentare* şi *partiționare* a tabelelor. Activitatea de *fragmentare* a datelor implică partiționarea bazei de date în fragmente disjuncte şi alocarea fiecărui fragment pe un singur nod (datele trebuie să fie stocate în locul în care acestea sunt utilizate mai frecvent). Avantajele acestei variante sunt costurile reduse cu stocarea şi comunicația. Cu toate acestea, disponibilitatea şi siguranța datelor sunt reduse, chiar dacă acestea sunt

**queries**

A global query can be expressed using different languages based on relational algebra or relational calculation.

**a). Tree representation** is one of the techniques most used to represent a relational algebraic expression. Query tree is a graphical representation of operations and operands that appear in an algebraic expression. The relations (operands) are represented in the leaf nodes (each node is an operation with one or more operands) and operations in the internal nodes. In the query execution, an internal node can be executed when its operands are available. After execution, the node is replaced with the result of the operation. The root node is the last executed and replaced with the final result.

**b). Query graph** will replace the query tree if relational calculation is used. The constants and attributes of the relation are represented in the graph nodes. It is allowed to repeat the attributes that appear in more than one relation. The conditions of junction are represented on the rows which connect the attribute nodes, and the conditions of selection are represented on the rows which connect a constant to an attribute.Citiți fonetic

## 5. Optimizing Queries

Conceptually, the query optimization consists of choosing the best strategies (order of the operations) of all the strategies possible.

In close relationship with distributed query optimization are the concepts of *fragmentation* and *partitioning* of tables. The activity of data *fragmentation* involves partitioning the database into disjoint fragments and allocating each fragment to a single node (the data must be stored on the location where those are used more frequently). The advantages of this variant are reduced costs for storage and communication. However, data availability

mai bune decât în cazul centralizării.

De exemplu, în *partiţionarea orizontală*, o tabelă relaţională poate fi descompusă, astfel încât unele înregistrări sunt localizate într-un site şi altele în altul. Fie o tabela G cu subseturile sale G1, G2, G3, distribuite în trei baze de date aflate ca locaţie geografică în trei oraşe distincte: New York, Paris, Los Angeles. O exemplificare practică a acestui tabel ar fi aceea în care ne imaginăm că aceasta este tabela cu angajaţii unei companii care lucrează în fiecare din aceste trei oraşe. Prin această partiţionare orizontală (pe linii) înregistrările angajaţilor care lucrează în Paris se vor afla în baza de date din Paris, îmbunătăţind astfel viteza de procesare dat fiind faptul că majoritatea accesărilor tabelei cu angajaţii din Paris se vor face de la sediul lor din Paris. Dezavantajul constă în faptul că atunci cand o aplicaţie necesită acces la toate înregistrările angajaţilor din toate oraşele în cadrul companiei, aceasta trebuie să colecteze datele de la fiecare dintre aceste noduri. În *partiţionarea verticală*, coloanele unui tabel sunt partiţionate între mai multe oraşe din reţea. Fiecare astfel de partiţionare trebuie să includă atributul cheie primară din tabel. De exemplu, tabelul cu salariile angajaţilor ar putea fi stocat într-un oraş în timp ce tabelul cu competenţele angajaţilor ar putea fi stocat în alt oraş. Dar o interogare concepută pentru a extrage informaţii diferite despre un anumit angajat va necesita şi informaţii localizate pe alte site-uri.

Problema optimizării execuţiei interogărilor distribuite are scopul de a obţine un timp de răspuns al sistemului cât mai mic posibil în condiţiile minimizării costului total de execuţie.

*Costul total de execuţie al unei interogăr*i într-un sistem distribuit constă din costul de prelucrare (CCPU), costurile de accesare a discurilor (CI/O) şi costul comunicaţiei între noduri. Costul comunicaţiei constă din costul de trimitere şi primire mesaje între noduri (CMSG) şi, respectiv, costul de transfer date între noduri (CTR). Costurile de comunicaţie depind în and security are reduced even if they are better than in the case of centralization.

For example, in *horizontal partitioning*, a relational table can be split so that some records are located in one site and others in another site. Let's assume a table G with its subsets G1, G2, G3, distributed in three databases located in three different cities: New York, Paris, Los Angeles. A practical example of this table would be to imagine that this is a table with the employees of a company working in each of these three cities. By this horizontal partitioning (on rows) records of employees who work in Paris will be in the database located on the node in Paris, thereby improving processing speed because most queries with workers in Paris will be made from Paris. The disadvantage is that when an application requires access to all records of all employees in the company, it must collect data from each of these three nodes. In *vertical partitioning*, the columns of a table are partitioned between several cities in the network. Each such partitioning must include the primary key attribute of the table. For example, employee salary table could be stored in a city while staff skills table could be stored in another city. But a query designed to receive various information about a particular employee will also need information from other sites.

The problem of optimizing distributed query execution has the goal to obtain a system response time as small as possible but also to minimize total cost of execution.

*The total cost of a query execution* in a distributed system consists of the processing cost (CCPU), the cost of accessing the discs (CI/O) and the cost of communication between nodes. The communication cost consists of the cost of sending and receiving messages between nodes (CMSG) and respectively the cost of transferring data between nodes (CTR). The communication costs depend largely of the type of communication network.

The formula of the total cost can be written as follows:

mare măsură de tipul reţelei de comunicaţie. Formula costului total poate fi scrisă, după cum urmează:

**Cost total = CCPU * nr_instrucţiuni + CI/O * nr_I/O + CMSG * nr_mesaje + CTR * cantitate_date**

(1)

*Timpul de răspuns al sistemului distribuit* se calculează din momentul în care interogarea începe până când primeşte răspuns din partea sistemului. O influenţă semnificativă în reducerea timpului de răspuns o au procesarea paralelă şi, respectiv, comunicaţia paralelă. Cu cât timpul de execuţie şi comunicaţie în paralel sunt mai mari, cu atât mai mic va fi timpul de răspuns. Formula timpului de răspuns al sistemului distribuit este următoarea:

**Timp răsp = TCPU * nr_instrucţiuni + TI/O * nr_I/O + TMSG * nr_mesaje + TTR * cantitate_date**

(2)

Reducerea la minimum a timpului de răspuns se realizează prin creşterea gradului de paralelism al execuţiei interogării distribuite. Acest lucru nu implică neapărat faptul că va fi minimizat şi costul total. Dimpotrivă, costul total poate creşte atunci când se încearcă să crească gradul de paralelism al execuţiei şi transmisiei. Pe de altă parte, minimizarea costului total implică îmbunătăţirea utilizării resurselor în detrimentul timpului de răspuns care va creşte. În practică, este de dorit un compromis între cele două obiective.

**Componentele optimizării cererilor distribuite**

- **Metoda de acces la date**. Cele mai multe SGBD-uri permit accesarea tabelelor în două moduri: *secvenţial* sau *indexat*. Cea mai bună metodă de acces la date poate fi determinată de contextul cererii. Este posibil să se returneze 90% din tuplurile tabelei, caz în care vom folosi accesul

**Total cost = CCPU * nr_instructions + CI/O * nr_I/O + CMSG * nr_messages + CTR * data_quantity**

**(1)**

*The response time of distributed system* is calculated from the moment when the query begins until it receives response from the system. A significant influence in minimizing response time has the parallel processing and respectively the parallel communication. The higher the execution and communication time the lower the response time will be. The formula of the distributed system response time is:

**Response time = TCPU * nr_ instructions + TI/O * nr_I/O + TMSG * nr_ messages + TTR * data_quantity**

**(2)**

The minimization of the response time is achieved by increasing the parallelism degree of a distributed query execution. This does not necessarily involve that the total cost will be minimized. On the contrary, the total cost may rise when trying to increase the degree of parallelism of the execution and transmission. On the other hand, minimizing the total cost involves better usage of resources in the detriment of the response time which will increase. In practice a compromise between those two objectives is desirable.

**The optimization of distributed queries components**

- **The data access method**. Most of the DBMS allow accessing the tables in two ways: sequential or indexed. The best method of data access can be determined by the query context. It is possible to return 90% of the tuples from the table, in which case we will use the sequential access. If the query will return 10% of the tuples, using an

secvențial. În cazul în care cererea va returna 10% din tuplurile tabelei, folosirea unui index ar creşte viteza de răspuns a sistemului.

- **Criteriile de uniune folosite**. Dacă interogarea accesează mai multe tabele cărora li se aplică operatorul de uniune, este determinant modul în care se face acest lucru. Optimizarea interogării trebuie să țină cont de ordinea în care uniunea tabelelor este făcută şi de numărul tuplurilor care iau parte la "join". Este, de asemenea, important pe care nod din rețea se va aplica operațiunea "join".
- **Costurile de comunicație**. În cazul în care datele de la mai multe noduri participă la operațiunea "join" cu scopul de a executa o interogare, trebuie să se ia în considerare costul transmiterii rezultatelor intermediare între noduri.

## 6. Factori care influențează optimizarea interogărilor distribuite

- **Securitatea sistemului** şi implicațiile autorizării accesului la date: permisiunile de accesare a datelor şi nodul pe care trebuiesc implementate rutinele de autorizare. Intercalarea rutinelor de autorizare în fluxul de execuție al operațiilor în sistem va creşte timpul de răspuns la interogări.
- **Disponibilitatea resurselor**. La un moment dat, este posibil ca unul dintre noduri să nu fie disponibil pentru interogare, sau chiar este posibil ca rețeua de comunicație să nu poată fi utilizată. Acest lucru creşte timpul de răspuns şi implicit costul total, sau chiar mai rău, poate face imposibilă execuția interogării dacă fragmentul care conține datele dorite nu este replicat la un alt nod disponibil în acel moment.
- **Constrângerile de integritate** declarate la nivelul bazei de date

index would increase system response speed.

- **The union criteria**. If the query accesses multiple tables which will use the union operator, it is very important the way in which this is done.
- The query optimization must take into account the order in which the union of the tables is done and the number of the tuples that take part to the "join". It is also important on which network node we will apply the operation "join".
- **The communication costs.** If the data from multiple nodes participates in the "join" operation in order to execute a query, we must consider the cost of transmitting intermediate results between nodes.

## 6. Factors that influence the optimization of distributed queries

- **The system security** and implications of data access authorization: what data someone can access and on what node the authorization routines should be implemented. The intercalation of the authorization routines in the execution flux of the operations in the system will increase its response time to queries.
- **The availability of resources**. At some point one of the nodes may not be available for querying, or the network may be down. This increases the response time and also the total cost, or even worse, may cause the query execution to fail if the fragment containing the desired data is not replicated at another node available at that time.
- **The integrity constraints** declared in the global database level help

globale ajută la optimizarea interogărilor adresate sistemului. În acest fel este posibil să eliminăm unele ramuri din arborele cererii datorită acestor constrângeri de integritate.

- **Implementarea şi modul de utilizare a reţelei de comunicaţie**. Traficul în reţea poate varia de la o zi la alta, chiar de la o oră la alta, ştiind că există momente în zi cu vârfuri de trafic în reţea şi, de asemenea, perioade cu trafic scăzut. Mai mult decât atât, pe măsură ce reţeaua suferă modificări de orice tip, optimizarea acesteia trebuie avută în vedere, fiind posibile eventual crearea de reţele de comunicaţie între noduri mult mai rapide.

Acestea sunt unele aspecte cu impact semnificativ asupra timpului de răspuns şi implicit a costului total de execuţie al unei interogări distribuite. De asemenea, cu cât numărul de factori care influenţează procesul de execuţie al cererilor creşte, cu atât optimizarea sistemului devine mai complexă.

## 7. Strategii de optimizare a execuţiei interogărilor

Există două strategii de optimizare a execuţiei interogărilor atunci când sunt necesare date stocate pe mai multe noduri: *query – site* şi *move – small*. Ideea este de a dezvolta o reprezentare grafică de dependenţă a fragmentelor independentă de site care să fie utilizată pentru a evidenţia dependenţele dintre fragmentele accesate de o interogare cu scopul de a formula şi rezolva problemele de alocare a datelor pentru sistemele de baze de date distribuite bazate pe (*query – site* şi *move – small*) strategii de execuţie a interogării. O soluţie optimă poate fi găsită atunci când strategia de execuţie *query – site* este utilizată, iar pentru strategia de execuţie *move – small* trebuie făcută o evaluare experimentală cu privire la eficacitatea unui algoritm euristic în realizarea unei soluţii aproape optime.

optimizing the queries addressed to the system. In this way it is possible to remove some branches from the query tree because of these integrity constraints.

- **The implementation and the usage of communication network**. Network traffic can vary from day to day, even from hour to hour, knowing that there are times of the day with network load peaks and also low traffic periods. Moreover, as the network undergoes changes of any kind, optimization solutions must be found.

Those are some aspects with significant impact on response time and hence the total cost of implementation of a distributed query. Also, while the number of factors that influence the performance of applications increases, the optimization of the system becomes very complex.

## 7. Strategies for optimizing the query execution

There are two strategies for optimizing the query execution when the necessary data is stored on multiple nodes: *queries - site* and *move-small.* The idea is to develop a site-independent fragment dependency graph representation to model the dependencies among the fragments accessed by a query, and use it to formulate and solve data allocation problems for distributed database systems based on (*query-site and move-small*) query execution strategies. An optimal solution can be achieved when the query-site execution strategy is employed, and for the move-small query execution strategy experimental evaluation must be performed about the effectiveness of a heuristic algorithm in achieving a near-optimal solution.

a). **The query – site strategy** states that if we have a query that uses data that are found in several fragments

**a). Strategia query – site** precizează că dacă avem o interogare care utilizează date folosite în mai multe fragmente situate pe noduri diferite, cea mai bună abordare ar fi să transferăm toate acele fragmente la nodul din care interogarea a fost iniţiată, şi să se execute acolo.

**b). Strategia move – small** precizează că dacă o operaţie binară implică două fragmente care sunt pe noduri diferite, atunci trebuie să transferăm cel mai mic fragment la nodul în care este situat fragmentul mai mare.

Este important să se ţină cont de aceste strategii, în scopul de a reduce la minimum cantitatea de date transferate între noduri, şi, în consecinţă, costul de comunicaţie, care reprezintă o parte semnificativă a costului total.

## Concluzii

Optimizarea interogărilor în sistemele distribuite este o activitate complexă care depinde de mulţi factori. Într-un anumit procent ea este realizată de către SGBD, dar există situaţii când aplicaţiile create trebuie să conţină şi algoritmi de optimizare a cererilor. Dacă fragmentarea bazei de date se face în mod corect, cele mai multe interogări se vor executa la nivel local. Pe de altă parte, politica de securitate a sistemului va restricţiona accesul utilizatorilor la o plajă definită de înregistrări, asupra cărora acţionează şi un set de constrângeri de integritate. Deşi putem avea o bază de date bine proiectată astfel încât proporţia prelucrărilor locale să fie foarte mare şi sortarea cererilor de către rutinele de securitate să fie foarte stricte, totuşi în decursul exploatării sistemului distribuit există nevoia execuţiei unor cereri mai complexe care să necesite date de la mai multe noduri. În funcţie de frecvenţa de execuţie a cererilor în sistem distribuit, este necesar să se găsească soluţia optimă pe baza procedurilor menţionate în acest document.

located on different nodes, the best approach would be to transfer all those fragments at the node where the query was initiated, and to execute it there.

b). **The move – small strategy states** that if a binary operation involves two fragments that are on different nodes, then we must transfer the smallest fragment to the node where the larger fragment is located.

It is important to keep in mind these strategies in order to minimize the quantity of data transferred between nodes, and consequently, the cost of communication, which is a significant part of the total cost.

## Conclusions

The optimization of queries in distributed systems is a complex activity that depends on many factors. In a certain percent it is performed by the DBMS, but there are situations when the user applications must contain algorithms for the query optimization. If fragmentation of the database is done correctly, most queries will run at local level. On the other hand, the system security policy will restrict the user's access to a well defined range of records, which also have a set of integrity constraints. Although we can have a well designed database so that the proportion of local processing should be very high and the sorting of queries based on security routines very strict, there is, however, the need of processing the execution of requests from distributed applications that require more complex data from multiple nodes. Depending on the frequency of execution of applications requests in the distributed system, it is necessary to find the optimal solution based on the procedures mentioned in this document.

## Bibliografie

1. Ozsu M.T., Valduriez P., *Principles of Distributed Database Systems*, Second Edition, Prentice Hall International, 1998.
2. Connolly T., Begg C., *Database Systems. A practical approach to design, implementation and management*, Fourth Edition, Addison Wesley, 2005.
3. Beynon-Davies P., *Database systems*, 3rd Edition, Palgrave-Macmillan, 2004.
4. Date C. J., *An introduction to database systems*, Eight edition, Pearson Education, 2004.

## Bibliography

1. Ozsu M.T., Valduriez P., *Principles of Distributed Database Systems*, Second Edition, Prentice Hall International, 1998.
2. Connolly T., Begg C., *Database Systems. A practical approach to design, implementation and management*, Fourth Edition, Addison Wesley, 2005.
3. Beynon-Davies P., *Database systems*, 3rd Edition, Palgrave-Macmillan, 2004.
4. Date C. J., *An introduction to database systems*, Eight edition, Pearson Education, 2004